数据结构与算法实习

A Simple Problem with Integers

> You have N integers, A1, A2, ... , AN. You need to deal with two kinds of operations. One type of operation is to add some given number to each number in a given interval. The other is to ask for the sum of numbers in a given interval.

对问题的分析设计过程

这道题涉及到了区间加法和区间求和，应该使用带有标记的线段树。当然也可以用两个树状数组来实现，以下将分析两种算法各自的优势。

程序中用到的数据结构和算法

- 线段树

线段树的标记有**标记下传**和**标记永久化**两种写法，本题中线段树并未嵌套其他数据结构，故使用标记下传的写法，每次访问一个节点首先将其上的ADD标记下传到其两个子节点上。使用了标记下传以后，区间上的SUM值就是真实的区间和了。

- 树状数组

考虑对数列$a_1, a_2, \ldots, a_n$进行差分

$$c_i = a_i - a_{i-1}$$

那么对于区间$[l, r]$的加法可以变为

$$c_l = c_l + ADD, c_{r+1} = c_{r+1} - ADD$$

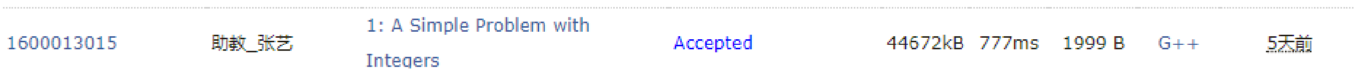
而

$$\sum_{i=1}^{k} a_i = \sum_{i=1}^{k}\sum_{j=1}^{i} c_j = \sum_{i=1}^{k} (k+1-i) * c_i = k * \sum_{i=1}^{k} c_i - \sum_{i=1}^{k} (i-1) * c_i$$

用树状数组分别维护c_i和$(i-1) * c_i$两个数列即可

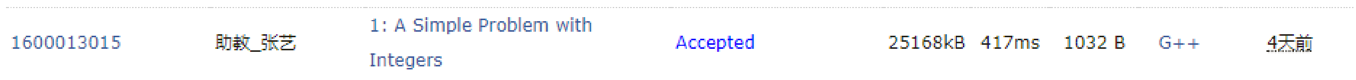
程序的运行情况

- 线段树



1600013015	助教_张艺	1: A Simple Problem with Integers	Accepted	44672kB	777ms	1999 B	G++	5天前

- 树状数组



1600013015	助教_张艺	1: A Simple Problem with Integers	Accepted	25168kB	417ms	1032 B	G++	4天前

在实习过程中得到的经验和体会

- 有些数据结构题目可以经过简单的数学变化，使得题目变得更加容易解决
- 对于满足区间减法的操作，树状数组往往可以代替线段树，代码短而且常数小

K-th Number

> You are working for Macrohard company in data structures department. After failing your previous task about key insertion you were asked to write a new data structure that would be able to return quickly k-th order statistics in the array segment. That is, given an array a[1...n] of different integer numbers, your program must answer a series of questions Q(i, j, k) in the form: "What would be the k-th number in a[i...j] segment, if this segment was sorted?" For example, consider the array a = (1, 5, 2, 6, 3, 7, 4). Let the question be Q(2, 5, 3). The segment a[2...5] is (5, 2, 6, 3). If we sort this segment, we get (2, 3, 5, 6), the third number is 5, and therefore the answer to the question is 5.

对问题的分析设计过程

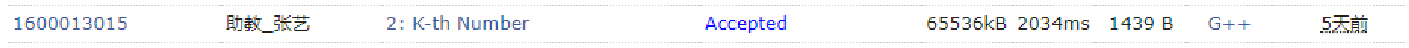
这是一个经典的区间第K大的问题，用线段树解决有三种常见的方法，分别是

- 可持久化线段树（主席树） 时间复杂度$O(n\log A + m\log A)$
- 树套树，外层是位置线段树、内层是权值线段树，或者外层是权值线段树，内层是位置线段树 时间复杂度都是$O(n\log^2 A + m\log^2 A)$
- 简单的位置线段树，每个节点按升序存$num_{[l,r]}$，每次查询二份答案 时间复杂度$O(n\log n + m\log A\log^2 n)$

程序中用到的数据结构和算法

这里选择使用主席树，主席树用n棵线段树$T_{[1,n]}$维护$num_{[1,i]}$对应的权值线段树。其中T_i以T_{i-1}为模板，通过新增一条从根节点到num_i的链构造。在查询的时候，通过对T_r和T_{l-1}两棵线段树做减法得到$num_{[l,r]}$对应的权值线段树，在这棵树上二分从而得到答案。

程序的运行情况



1600013015　助教_张艺　2: K-th Number　Accepted　65536kB 2034ms 1439 B　G++　5天前

在实习过程中得到的经验和体会

同一道题目往往有很多不同的做法，我们应该根据题目的要求选择一个最合适的方法

Lost Cows

> N (2 <= N <= 8,000) cows have unique brands in the range 1..N. In a spectacular display of poor judgment, they visited the neighborhood 'watering hole' and drank a few too many beers before dinner.

> When it was time to line up for their evening meal, they did not line up in the required ascending numerical order of their brands.
>
> Regrettably, FJ does not have a way to sort them. Furthermore, he's not very good at observing problems. Instead of writing down each cow's brand, he determined a rather silly statistic: For each cow in line, he knows the number of cows that precede that cow in line that do, in fact, have smaller brands than that cow.
>
> Given this data, tell FJ the exact ordering of the cows.

对问题的分析设计过程

K_i表示第i个数字是$num_{[1,i]}$中第K_i+1大的数字，若我们知道了$num_{[i+1,n]}$是那些数字，也就知道了$num_{[1,i]}$有哪些数字了。从后往前依次确定num_i，只需维护还剩下哪些数字可以选取就好了。考虑到数据不是很大，可以用线段树或者vector维护。

程序中用到的数据结构和算法

- 线段树

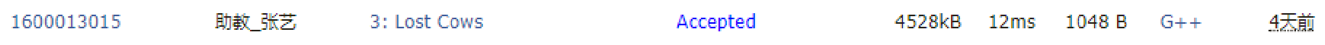
用线段树维护区间和，最开始建立一个叶节点全部为1的线段树。每用过一个数字，就在对应的叶节点加上-1，查询第K大在线段树上面二分即可。

- vector

首先将$[1,n]$依次插入vector。每用过一个数字，就将其暴力删除，查询第K大就是vector中第K个元素。

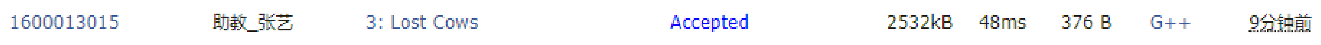
程序的运行情况

- 线段树



1600013015	助教_张艺	3: Lost Cows	Accepted	4528kB	12ms	1048 B	G++	4天前

- vector



1600013015	助教_张艺	3: Lost Cows	Accepted	2532kB	48ms	376 B	G++	9分钟前

在实习过程中得到的经验和体会

- 对于数据不是很大的题目，如果暴力可以通过则没有必要写最优解
- 熟练使用STL可以提高编程效率

代码

> A Simple Problem with Integers By 线段树

```
#include <cstdio>
#include <cstdlib>
```

```
using namespace std;

const int MAXN = 1000000;

typedef long long LL;

int nodeTotel = 1, val[MAXN];

struct TreeNode{
    LL sum, add;
           int l, r;
    TreeNode *ls, *rs;

    TreeNode() {
        sum = add = 0;
    }

    void update() {
        sum = ls->sum + rs->sum;
        sum = sum + (r - l + 1) * add;
    }

    void download() {
        ls->add += add;
        rs->add += add;
               ls->sum += (ls->r - ls->l + 1) * add;
               rs->sum += (rs->r - rs->l + 1) * add;

        add = 0;
    }
} node[MAXN], *root, *null;

void init() {
    null = &node[0];

    null->ls = null->rs = null;
}

TreeNode *treeBuild(int l, int r) {
    TreeNode *curr = &node[nodeTotel++];

    curr->ls = curr->rs = null;
    curr->l = l, curr->r = r;

    int mid = l + (r - l) / 2;

    if (l == r)
        curr->sum = val[mid];
    else {
        curr->ls = treeBuild(l, mid);
        curr->rs = treeBuild(mid + 1, r);

        curr->update();
    }

    return curr;
}

void treeAdd(TreeNode *curr, int L, int R, int x) {
    int l = curr->l, r = curr->r;
    curr->download();

    if (l >= L && r <= R) {
        curr->add = x;
        curr->sum += (r - l + 1) * x;
    }
    else {
        int mid = l + (r - l) / 2;

        if (L <= mid)
            treeAdd(curr->ls, L, R, x);
        if (R >= mid + 1)
            treeAdd(curr->rs, L, R, x);

        curr->update();
    }
}

LL treeQuery(TreeNode *curr, int L, int R) {
    int l = curr->l, r = curr->r;

    curr->download();

    LL ans = 0;

    if (l >= L && r <= R)
        ans = curr->sum;
    else {
        int mid = l + (r - l) / 2;

        if (L <= mid)
            ans += treeQuery(curr->ls, L, R);
        if (R >= mid + 1)
            ans += treeQuery(curr->rs, L, R);
    }

    return ans;
}

int main() {
    init();

    int n; scanf("%d", &n);
    int m; scanf("%d", &m);

    for (int i = 1; i <= n; i++)
        scanf("%d", &val[i]);

    root = treeBuild(1, n);

    for (int i = 1; i <= m; i++) {
        char ope[10]; scanf("%s", ope);

        int l; scanf("%d", &l);
        int r; scanf("%d", &r);

        if (ope[0] == 'C') {
            int x; scanf("%d", &x);
            treeAdd(root, l, r, x);
        }
        if (ope[0] == 'Q')
            printf("%lld\n", treeQuery(root, l, r));
    }

    return 0;
}
```

> A Simple Problem with Integers By 树状数组

```
#include <cstdio>

typedef long long LL;

const int MAXN = 100010;

LL c1[MAXN], c2[MAXN];

int num[MAXN];

void Insert(LL *c, int x, LL y) {
    for (; x < MAXN; x += x & -x)
        c[x] += y;
}

LL Query(LL *c, int x) {
    LL ans = 0;
    for (; x; x -= x & -x)
        ans += c[x];
    return ans;
}

int main() {
    int n; scanf("%d", &n);
    int m; scanf("%d", &m);

    for (int i = 1; i <= n; i++)
        scanf("%d", &num[i]);

    for (int i = n; i >= 1; i--)
        num[i] -= num[i - 1];

    for (int i = 1; i <= n; i++) {
        Insert(c1, i, num[i]);
        Insert(c2, i, 1ll * num[i] * (i - 1));
    }

    for (int i = 1; i <= m; i++) {
        char o[10]; scanf("%s", o);

        int l; scanf("%d", &l);
        int r; scanf("%d", &r);

        if (o[0] == 'Q') {
            l -= 1;

            LL sum1 = r * Query(c1, r) - Query(c2, r);
            LL sum2 = l * Query(c1, l) - Query(c2, l);

            printf("%lld\n", sum1 - sum2);
        }

        if (o[0] == 'C') {
            r += 1;

            int x; scanf("%d", &x);

            Insert(c1, l, x), Insert(c2, l, (l - 1) * x);
            x = -x;
            Insert(c1, r, x), Insert(c2, r, (r - 1) * x);
        }
    }

    return 0;
}
```

> K-th Number By 主席树

```
#include <cstdio>
#include <cstdlib>
#include <climits>

using namespace std;

const int MAXN = 5000000;

struct TreeNode {
    int totel;

    TreeNode *ls, *rs;
} node[MAXN], *tree[MAXN], *null;

int nodeTotel = 1;

TreeNode *treeInsert(TreeNode *prev, int l, int r, int val) {
    TreeNode *curr = &node[nodeTotel++];
    curr->ls = curr->rs = null;

    curr->totel = prev->totel + 1;

    if (l != r) {
        int mid = l + (r - l) / 2;

        if (val <= mid) {
            curr->ls = treeInsert(prev->ls, l, mid, val);
            curr->rs = prev->rs;
        }

        else {
            curr->ls = prev->ls;
            curr->rs = treeInsert(prev->rs, mid + 1, r, val);
        }
    }

    return curr;
}

int treeQuery(TreeNode *head, TreeNode *tail, int l, int r, int k) {
    int temp = tail->ls->totel - head->ls->totel,
        mid = l + (r - l) / 2;

    if (l == r)
        return mid;

    if (k <= temp)
        return treeQuery(head->ls, tail->ls, l, mid, k);
    else
        return treeQuery(head->rs, tail->rs, mid + 1, r, k - temp);
}

void init() {
    null = &node[0];

    null->ls = null->rs = null;
    null->totel = 0;

    tree[0] = null;
}

const int INTMAX = 1000000000;

int main() {
    init();

    int n, m;
    scanf("%d %d", &n, &m);

    for (int i = 1; i <= n; i++) {
        int v;
        scanf("%d", &v);

        tree[i] = treeInsert(tree[i - 1], -INTMAX, INTMAX, v);
    }

    for (int i = 1; i <= m; i++) {
        int l, r, k, ans;
        scanf("%d%d%d", &l, &r, &k);

        ans = treeQuery(tree[l - 1], tree[r], -INTMAX, INTMAX, k);
        printf("%d\n", ans);
    }

    return 0;
}
```

> Lost Cows By 线段树

```
#include <cstdio>
#include <cstdlib>

using namespace std;

const int MAXN = 1000000;

int val[MAXN], num[MAXN], sum[MAXN];

int Query(int c, int l, int r, int k) {
    int mid = (l + r) / 2;
    if (l == r)
        return mid;

    int curr = sum[2 * c];

    if (curr >= k)
        return Query(2 * c, l, mid, k);
    else
        return Query(2 * c + 1, mid + 1, r, k - curr);
}

void Insert(int c, int l, int r, int x) {
    int mid = (l + r) / 2;

    sum[c] -= 1;

    if (l == r)
        return;

    if (x <= mid)
        Insert(2 * c, l, mid, x);
    else
        Insert(2 * c + 1, mid + 1, r, x);
}

void Build(int c, int l, int r) {
    if (l == r)
        sum[c] = 1;
    else {
        int mid = (l + r) / 2;

        Build(2 * c, l , mid);
        Build(2 * c + 1, mid + 1, r);

        sum[c] = sum[2 * c] + sum[2 * c+ 1];
    }
}

int main(int argc, char** argv)  {
    int n; scanf("%d", &n);

    Build(1, 1, n);

    for (int i = 2; i <= n; i++)
        scanf("%d", &val[i]);

    for (int i = n; i >= 1; i--) {
        num[i] = Query(1, 1, n, val[i] + 1);
        Insert(1, 1, n, num[i]);
    }

    for (int i = 1; i <= n; i++)
        printf("%d\n", num[i]);

    return 0;
}
```

Lost Cows By 暴力

```cpp
#include <iostream>
#include <vector>

using namespace std;

vector<int> a;

int b[1000000], c[1000000];

int main() {
    int n; cin >> n;

    for (int i = 1; i <= n; i++)
        a.push_back(i);

    for (int i = 2; i <= n; i++)
        cin >> b[i];

    for (int i = n; i >= 1; i--)
        c[i] = a[b[i]], a.erase(a.begin() + b[i]);

    for (int i = 1; i <= n; i++)
        cout << c[i] << endl;

    return 0;
}
```